CS422 - Programming Language Design

K: A Rewrite Logic Framework for Programming Language Design

Grigore Roşu

Department of Computer Science
University of Illinois at Urbana-Champaign

What is K

K is an algebraic framework for defining programming languages or type systems for them. It consists of

- The *K-technique*, which is based on rewriting *modulo* associativity, commutativity and identity, and uses a first-order representation of *continuations*;
- The *K-notation*, which consists of a series of conventions that make the language definitions intuitive, easy to understand, to read and to teach, compact, modular and scalable.

The K framework is introduced by defining λ_K, a simple higher-order programming language. From here on in the class we'll use K to define languages or language features.

Why K

As seen in previous lectures, the *SOS definitional styles were problematic* when we wanted to define more complex features of languages, such as a halt statement, regardless of whether they are used on paper or formalized and automated using rewriting.

We have also seen that the rewrite-logic-based *functional style had the same limitations as big-step SOS*.

The *continuation style appears to be more appropriate* to define complex languages, because one has the control context of the program to evaluate explicit in the state, so one could easily define statements that change that control.

However, *the continuation style looked rather heavy* and hard to read and understand, at least in its Maude representation.

K builds upon the continuation style, providing a highly optimized notation and several intuitive conventions.

K can be mechanically translated into rewriting logic, so our K-language definitions can be executed in Maude after translation.

We exemplify the K framework by defining a simple functional language; yet, this language would be difficult to define if one does not use a good definitional formalism. Basic principles of functional programming were explained in the previous lecture notes.

Syntax of λ_{K}

$$
\begin{array}{rcl}
\mathit{Var} & ::= & \text{identifier} \\
\mathit{Bool} & ::= & \text{assumed defined, together with } \mathsf{not}_{\mathit{Bool}}\_ \colon \mathit{Bool} \to \mathit{Bool}, \text{etc.} \\
\mathit{Int} & ::= & \text{assumed, together with basic operations such as} \\
 & & \_ +_{\mathit{Int}} \_ \colon \mathit{Int} \times \mathit{Int} \to \mathit{Int},\ \_ <_{\mathit{Int}} \_ \ :\ \ \mathit{Int} \times \mathit{Int} \to \mathit{Bool}, \text{etc.} \\
\mathit{Exp} & ::= & \mathit{Var} \mid \mathit{Bool} \mid \mathit{Int} \mid \mathsf{not}\ \mathit{Exp} \mid \mathit{Exp} + \mathit{Exp} \mid \mathit{Exp} < \mathit{Exp} \mid \ldots \\
 & & \mid \lambda\, \mathit{VarList}^{[,]}.\mathit{Exp} \mid \mathit{Exp}\ \mathit{ExpList}^{[,]} \\
 & & \mid \mathsf{if}\ \mathit{Exp}\ \mathsf{then}\ \mathit{Exp}\ \mathsf{else}\ \mathit{Exp} \\
 & & \mid \mathsf{ref}\ \mathit{Exp} \mid\ *\mathit{Exp} \mid \mathit{Exp} := \mathit{Exp} \\
 & & \mid \mathsf{halt}\ \mathit{Exp}
\end{array}
$$

Other common functional language constructs can be easily defined using the constructs above, as *syntactic sugar*:

- let $X = E$ in E' is $(\lambda X.E')E$,
- let $F(X) = E$ in E' is $(\lambda F.E')(\lambda X.E)$,
- let $F(X, Y) = E$ in E' is $(\lambda F.E')(\lambda X, Y.E)$, etc., and
- $E; E'$ is $(\lambda D.E')E$, where D is a fresh "dummy" variable.

With these, if n is for example 3, then

$$\begin{array}{l}
\mathsf{let}\ r = \mathsf{ref}\ n \\
\mathsf{in\ let}\ g(m,h) = \ \mathsf{if}\ m > 1\ \mathsf{then}\ (r := (*r) * m;\ h(m-1,h))\ \mathsf{else\ halt}\ (*r) \\
\quad \mathsf{in}\ g(n-1,g)
\end{array}$$

translates into

$$\begin{array}{l}
(\lambda r\ . \\
\quad (\lambda g\ .\ g(3-1,g)) \\
\quad (\lambda m,h\ .\ \mathsf{if}\ m > 1 \\
\qquad\qquad \mathsf{then}\ (\lambda d\ .\ h(m-1,h)\ (r := (*r) * m)) \\
\qquad\qquad \mathsf{else\ halt}\ (*r) \\
\quad) \\
)\ (\mathsf{ref}\ 3)
\end{array}$$

Definition of λ_{K} in K

The complete K definition of λ_{K} is shown in Figure 1 in the K-report (a link to it is provided). We discussed this definition and K in class; a more detailed discussion can be found in the report.

Exercise 1 *Read the first 22 pages (the introduction) of the K-report.*